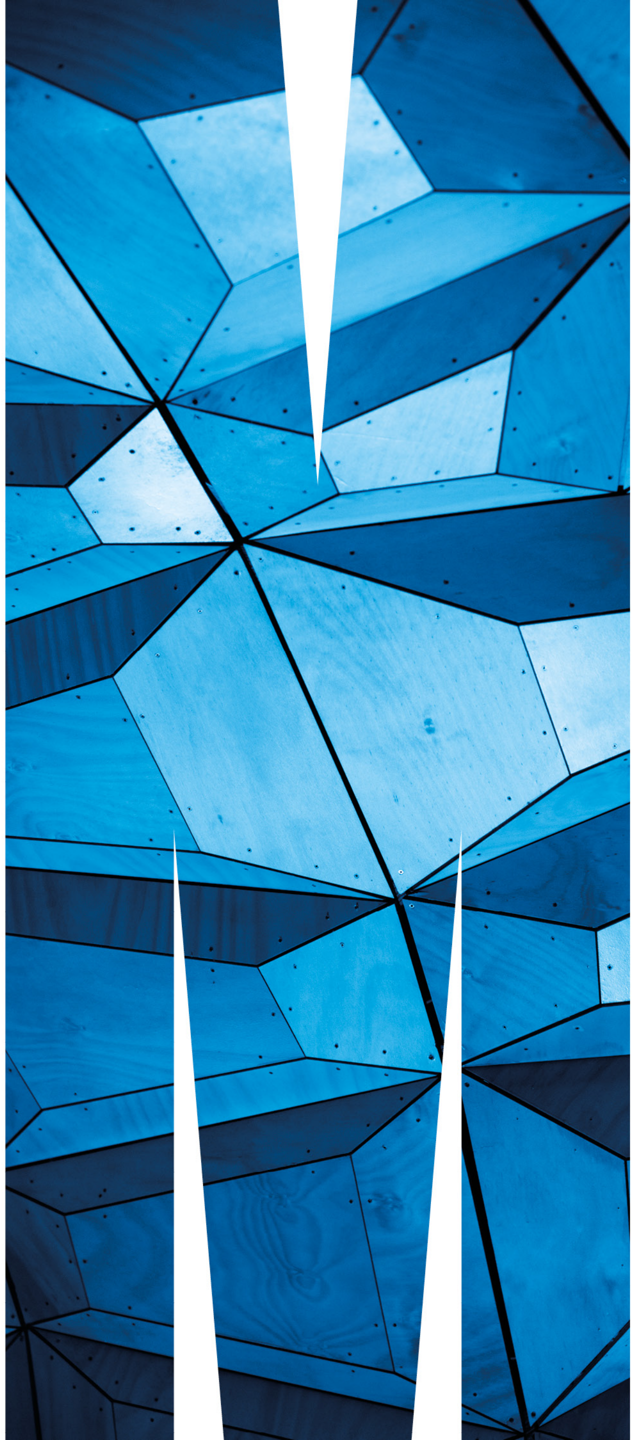
MONASH University

Operating systems (continued)



Interprocess communication

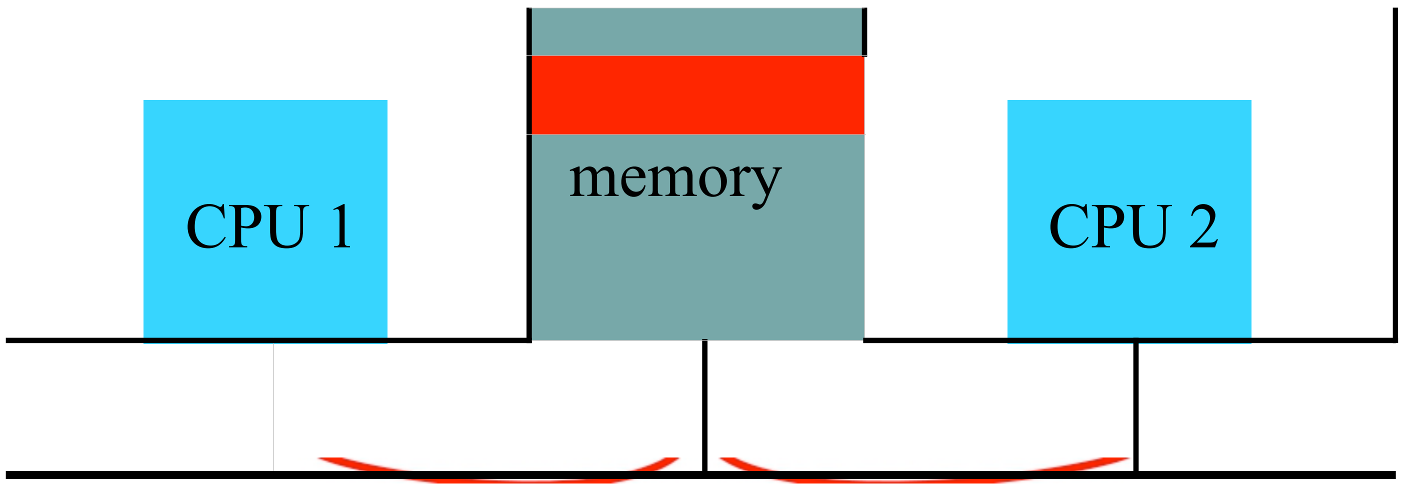
- Interprocess communication (IPC): OS provides mechanisms so that processes can pass data.
- Two types of semantics:
  - blocking : sending process waits for response;
  - non -blocking : sending process continues.

IPC styles/techniques

- Shared memory:
  - processes have some memory in common;
  - must cooperate to avoid destroying/missing messages.
- Message passing:
  - processes send messages along a communication channel ---no common address space.

Shared memory

- Shared memory on a bus:



Race condition in shared memory

- Problem when two CPUs try to write the same location:
  - CPU 1 reads flag and sees 0.
  - CPU 2 reads flag and sees 0.
  - CPU 1 sets flag to one and writes location.
  - CPU 2 sets flag to one and overwrites location.

Atomic test -and -set

- Problem can be solved with an atomic test - and - set:
  - single bus operation reads memory location, tests it, writes it.
- NIOS- II provides a hardware implementation of an atomic test and set instruction for mutual exclusion implementation in multi - processor system.
- ARM test - and - set provided by SWP - indivisible swap instruction:

```
        ADR r0, GATEKEEPER     ; r0 gets memory address
        LDR r1,#1              ; constant 1 in r1
GETFLAG: SWP r1,r1,[r0]        ; r1 < -> GATEKEEPER mem
        BNZ GETFLAG  ; Zero flag set from memory.  Branch Not Z
                     ;If someone beat us (ie already 1), try again
                     ; ie wait for zero to be written when other CPU leaves
; read or write to shared memory here …………….
      LDR r1, #0          ; finish access by
      SWP r1, r1, [r0]    ; writing 0 to GATEKEEPER so others allowed access
```

This material has been adapted from chapter 6 slides provided with the textbook by Wolf "Computers as Components: principles of embedded system design" Morgan Kaufmann © .

Critical regions

- Critical region : section of code that cannot be interrupted by another process.
- Examples:
  - writing shared memory;
  - accessing I/O device.
- Looking forward?
  - Goto Mutex notes
  - Example3_IncDecCS

Semaphores

- Semaphore: OS primitive for controlling access to critical regions and coordinating multiple processes.
- Protocol:
  - Get access to semaphore with Wait() - called OSSemPend() in uC/OS -II.
  - Perform critical region operations.
  - Release semaphore with Signal() – called OSSemPost() in uC/OS -II.

Invented by Edsger Dijkstra in 1962 and in many references still called PV semaphores taken from the Dutch words for Wait and Signal

Semaphore Definition

- A semaphore consists of its own:
- Counter (unsigned integer)
- Wait function eg OSSemPend()
- Signal function eg OSSemPost()
- Queue of blocked processes waiting for a signal to arrive from another process.

Semaphore Semantics

Waiting on a semaphore:

- If (counter > 0) decrements the counter .
- If (counter = 0) block and add caller to queue of blocked processes

Signalling a semaphore reverses a wait:

- If the queue is empty the counter increments.
- Otherwise the highest priority queued process is made ready to run – counter remains 0.

Semaphore Analogy

- Processes < - > people
- Semaphore < -> meeting place (eg park bench) with a jar of tokens
- Counter < -> number of tokens in jar.
- Wait <-> a person picks up a token or waits in a queue at the bench for a token to arrive.
- Signal < -> person deposits a token and moves away.

Semaphore Synchronisation

- Event signalling
- Synchronising interprocess data transfer
- Mutual exclusion of processing in critical sections.

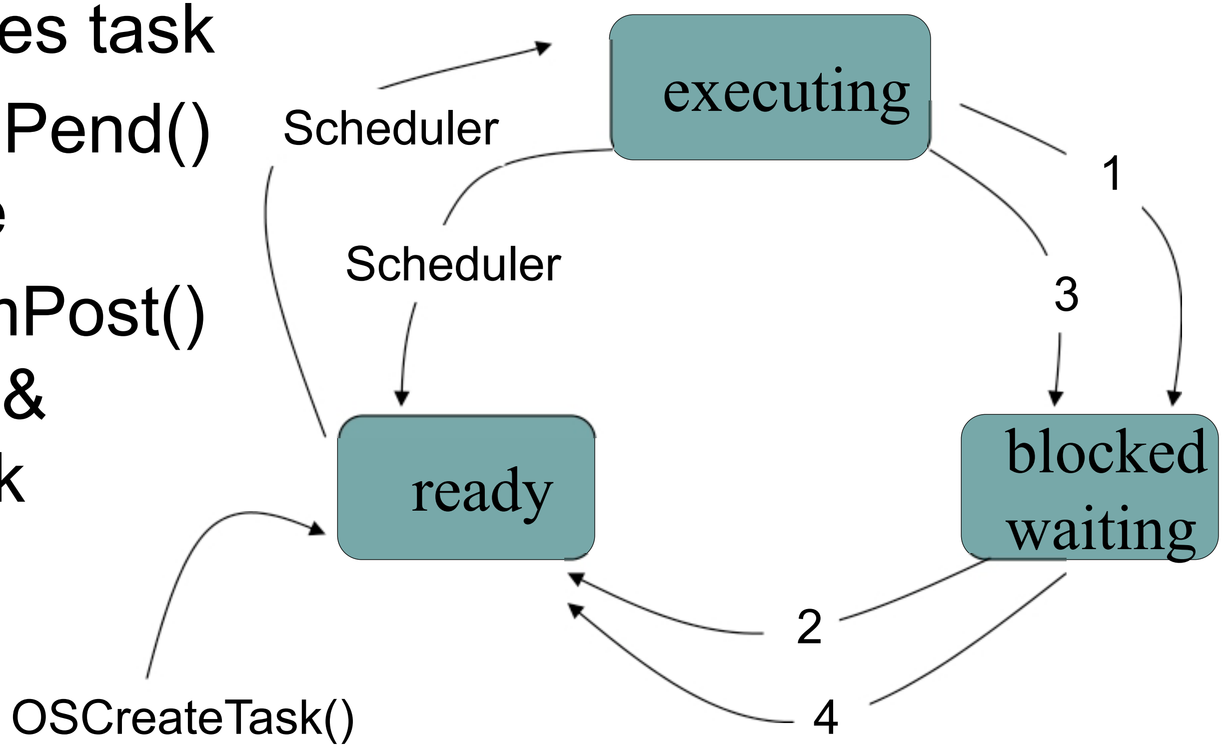
Process States with Semaphores

1 Sleep = OSTimeDly()

2 Timer ISR wakes task

3 Wait = OSSemPend() on 0 semaphore

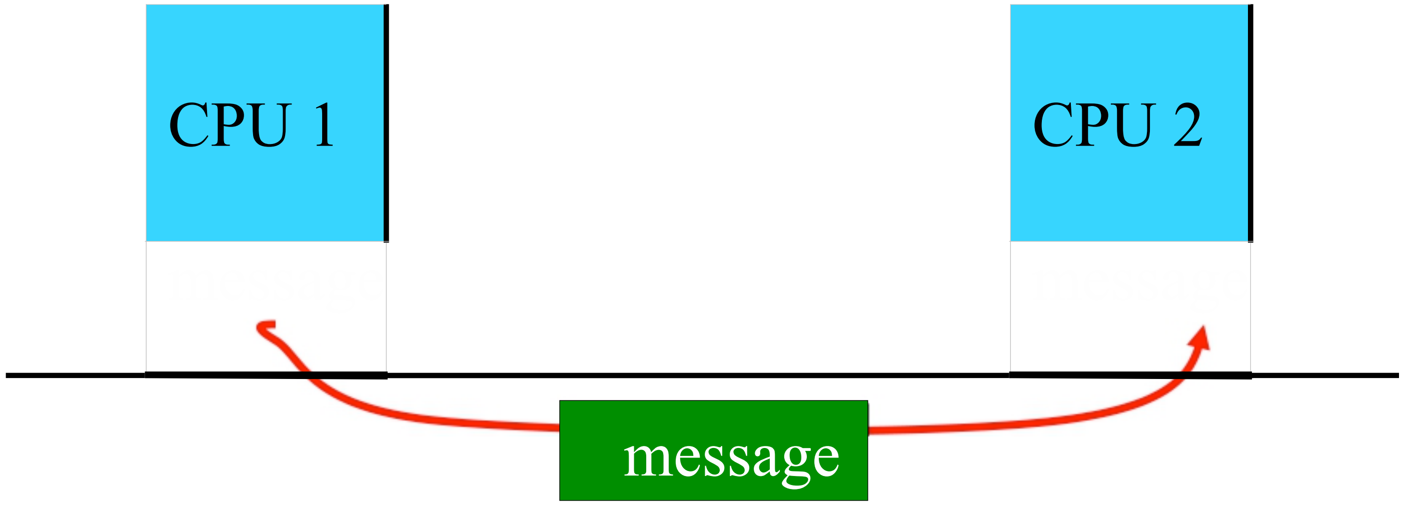
4 Signal=OSSemPost() by a (different & executing) task



This material has been adapted from chapter 6 slides provided with the textbook by Wolf "Computers as Components: principles of embedded system design" Morgan Kaufmann © .

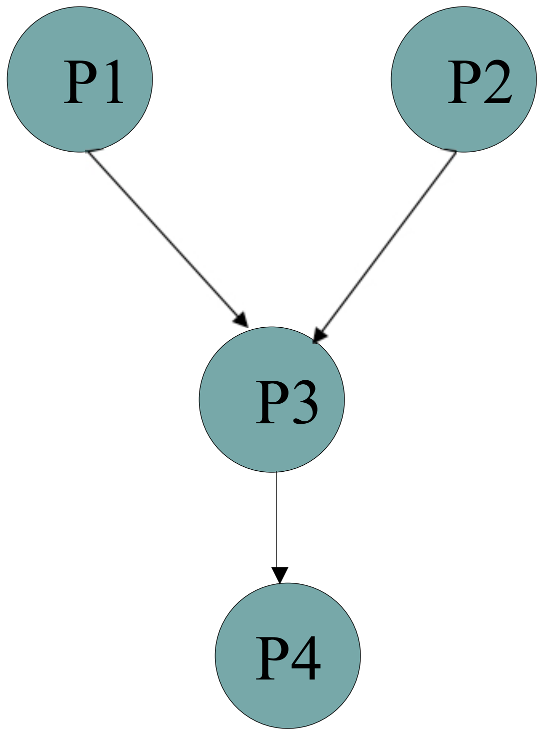
Message passing

- Message passing on a network:



Process data dependencies

- One process may not be able to start until another finishes.
- Data dependencies defined in a task graph .
- All processes in one task run at the same rate.



This material has been adapted from chapter 6 slides provided with the textbook by Wolf “Computers as Components: principles of embedded system design” Morgan Kaufmann © .

Other operating system functions

- Date/time.
- File system.
- Networking.
- Security.